О программе “Репетитор”

Д.В. Хмелёв

3 июля 2002

Аннотация

Описание программы “Репетитор” и её алгоритма.

Содержание

1 Введение

В июне 2002 года у *зарегистрированных авторовw* Самиздата (`http://zhurnal.lib.ru/`) появилась новая возможность анализа своих текстов: программа Репетитор, которая находит все повторяющиеся подстроки в тексте. В дальнейшем мы будем называть повторяющуюся строку просто *повторением*.

Имеется разумное ограничение на длину повторения: 10 для поэтических текстов и 20 для прозы. Более того, для того, чтобы исключить вывод повторяющихся длинных слов вроде “обороноспособности” для поэзии введено дополнительное ограничение: если имеется повторение длины от 10 до 15, то оно выводится только при наличии двух пробелов, разделяющих слова. Такое ограничеие позволяет отлавливать повторения вроде “который сказал” и пр.

Текст предварительно преобразуется в “абзацы”, которые определяются с помощью специального конечного автомата. Новый абзац начинается после пустой строки, или знака препинания, за которым следует перевод строки и пробельный символ. Возможны некоторые отступления от указанного правила. Программа не ищет повторения, пересекающие границу абзаца.

Максимальное количество выдаваемых повторений составляет 3000.

Время работы и требуемая память пропорциональны длине входного текста. В следующем разделе дано описание алгоритма работы программы.

2 Описание алгоритма

Начнём с простейших определений [**?**]. *Строкой* S называется упорядоченный список букв, записанных последовательно слева направо. Под $S[i..j]$ подразумевается *подстрока* строки S, которая начинается с символа с номером i и заканчивается номером j. *Длина* строки S обозначается через$|S|$. Подстрока $S[1..i]$ называется *префиксом* строки S, а подстрока $S[i..|S|]$ называется *суффиксом* строки S.

Например, "Мы все учились понемногу" — это строка, "все учились " — подстрока, "Мы все" — префикс, "понемногу" — суффикс.

$S(i)$ обозначает символ i строки S.

На входе алгоритм получает строку S длины n. Прежде, чем описывать алгоритм необходимо более строго сказать, что имеется ввиду под "повторением". Точнее, нам следует сначала определить, что какие строки образуют "повторяющуюся пару". Неправильный выбор определения "повторяющейся пары" может привести к бессмысленно повторяющимся результатам. Например, если вся строка состоит из n повторения строки n, то алгоритм, который ищет "все пары повторяющихся подстрок", выдаст порядка n^4 пар, что совершенно нежелательно. Поэтому необходимо такое определение, которое каким-то образом отражало бы "максимальность" повторяющейся пары.

Максимальная пара (или максимальная повторяющаяся пара) в строке S есть пара таких одинаковых подстрок α и β, что буквы справа (и слева) от α и β различны. Таким образом, попытка расширить пару направо (налево) приведёт к её разрушению.

Максимальное повторение α есть подстрока S, которая встречается в какой-нибудь повторяющейся паре.

Репетитор ищет все максимальные повторения с использованием *суффиксного дерева*, с которого мы и начнём.

2.1 Суффиксное дерево

Напомним, что деревом называется неориентированный граф без циклов.

Суффиксным деревом T для подстроки S длины n называется дерево с корнем, у которого ровно n листьев, занумерованных числами от 1 до n. У каждой внутренней вершины, отличной от корня и называемой в дальнейшем *узлом*, имеется не меньше двух потомков, а каждое ребро помечено непустой подстрокой S. Метки разных рёбер, исходящих из любого узла (включая корень), начинаются с разных букв. Основной особенностью суффиксного дерева является то, что для любого листа i конкатенация строк на пути от корня к листу i в точности совпадает с суффиксом $S[i..n]$.

Существуют эффективные способы построения суффиксного дерева за время *пропорциональное* длине входного текста с использованием памяти, также пропорциональной длине входного текста [**?**]. Число узлов (а следовательно, и вершин) в суффиксном дереве пропорционально длине входного текста.

2.2 Максимальных повторений

Используя суффиксное дерево, можно найти все максимальные повторения за время, пропорциональное длине строки. Следующая лемма [**?**] даёт необходимое условие для того, чтобы какая-либо строка была максимальным повторением.

Лемма 2.1. *Пусть T — суффиксное дерево строки S. Если строка α является максимальным повторением, то она является конкатенацией меток рёбер, ведущик к какому-нибудь узлу v в дереве T (узел — это не лист!)*

Доказательство. Если α — максимальное повторение, то оно встречается в S по меньшей мере дважды, причём буква справа от первого употребления α отличается от буквы справа во втором употреблении α. Из определения суффиксного дерева вытекает, что α является меткой пути, ведущей к какому-нибудь узлу. □

Наложив простое дополнительное требование, можно получить необходимый и достаточный признак максимального повторения.

Для каждой позиции i в строке S обозначим через $S(i-1)$ *левую букву* суффикса $S[i..n]$. *Левая буква листа j* в дереве T есть левая буква суффикса $S[j..n]$.

Узел v дерева T называется *леворазнообразным* если хотя бы два листа в поддереве v отличаются левыми буквами. По определению, листья не являются леворазнообразными.

Заметим, что свойство леворазнообразности распространяется наверх. Если узел v леворазнообразен, то все его предки леворазнообразны.

Теорема 2.2 ([?]). *Строка α, являющаяся меткой пути к узлу v в дереве T является максимальным повторением тогда и только тогда, когда v леворазнообразен.*

2.3 Нахождение всех леворазнообразных узлов за линейное время

Для каждого узла v в дереве T алгоритм либо записывает, что узел v леворазнообразен, либо он записывает букву, которую мы обозначим x, которая является левой буквой всех листов в поддереве v. Вначале алгоритм записывает левые буквы для всех листов суффиксного дерева T. Далее он обходит дерево снизу вверх. Прежде обработки узла v, он рекурсивно вызывается для каждого из потомков

v. Если какой-нибудь из потомков оказывается леворазнообразным, то и v — леворазнообразен. Елси ни один из потомков v не леворазнообразен, то сравниваются записанные ранее буквы непосредственных потомков v. Если все они равны, скажем, x, то алгоритм записывает x на узле v. Если же встретились разные буквы, то алгоритм записывает, что v леворазнообразен. Время для проверки, что все сыновья v имеют одинаковую записанную букву, пропорционально их числу, которое не превосходит числа букв в алфавите: а поскольку число узлов линейно по n алгоритм линеен по n.

2.4 Об упрощениях в изложении алгоритма

На самом деле Репетитор ищет повторения для *набора* абзацев-строк $S_1, \ldots,$ S_n и строит суффиксное дерево для всех строк. Это приводит к незначительным усложнениям алгоритма.

Список литературы

[1] D.Gusfield, *Algorithms on strings, trees, and sequences.* Cambridge University Press, Cambridge, 1997.